UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/754,890	01/05/2001	K. Rustan M. Leino	9772-0274-999	4747

22879 7590 12/03/2004

HEWLETT PACKARD COMPANY
P O BOX 272400, 3404 E. HARMONY ROAD
INTELLECTUAL PROPERTY ADMINISTRATION
FORT COLLINS, CO 80527-2400

EXAMINER	
YIGDALL, MICHAEL J	

ART UNIT	PAPER NUMBER
2122	

DATE MAILED: 12/03/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

COMMISSIONER FOR PATENTS
UNITED STATES PATENT AND TRADEMARK OFFICE
P.O. BOX 1450
ALEXANDRIA, VA 22313-1450
www.uspto.gov

MAILED

DEC 0 3 2004

Technology Center 2100

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/754,890
Filing Date: January 05, 2001
Appellant(s): LEINO ET AL.

Philip S. Lyren
For Appellant

EXAMINER'S ANSWER

This is in response to the appeal brief filed August 20, 2004.

(1) Real Party in Interest

A statement identifying the real party in interest is contained in the brief.

(2) Related Appeals and Interferences

A statement identifying the related appeals and interferences which will directly affect or be directly affected by or have a bearing on the decision in the pending appeal is contained in the brief.

(3) Status of Claims

The statement of the status of the claims contained in the brief is correct.

(4) Status of Amendments After Final

The appellant's statement of the status of amendments after final rejection contained in the brief is correct.

(5) Summary of Invention

The summary of invention contained in the brief is correct.

(6) Issues

The appellant's statement of the issues in the brief is correct.

(7) Grouping of Claims

Appellant's brief includes a statement that claims 1-47 do not stand or fall together and provides reasons as set forth in 37 CFR 1.192(c)(7) and (c)(8).

(8) Claims Appealed

The copy of the appealed claims contained in the Appendix to the brief is correct.

(9) Prior Art of Record

Detlefs, David L. et al., "Extended Static Checking," Compaq Systems Research Center, SRC Research Report #159 (December 18, 1998), 44 pages.

4,920,538	Chan et al.	4-1990
5,854,924	Rickel et al.	12-1998

(10) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, 45 and 47 stand finally rejected under 35 U.S.C. 103(a) as being unpatentable over "Extended Static Checking" by Detlefs et al. (hereinafter "Detlefs") in view of U.S. Pat. No. 4,920,538 to Chan et al. (hereinafter "Chan").

Claims 6, 7, 9, 10, 12, 13, 15, 16, 18, 19, 21, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44 and 46 stand finally rejected under 35 U.S.C. 103(a) as being unpatentable over Detlefs in view of Chan and further in view of U.S. Pat. No. 5,854,924 to Rickel et al. (hereinafter "Rickel").

The rejection of claims 1-47 under 35 U.S.C. 103(a) is set forth in a prior Office action, mailed on May 17, 2004. The rejection is reproduced below, augmented to address the issues now presented:

Claim Rejections - 35 USC § 103

1. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

2. Claims 1-5, 8, 11, 14, 17, 20, 22-27, 30, 33, 36, 39, 42, 45 and 47 are rejected under 35 U.S.C. 103(a) as being unpatentable over Detlefs in view of Chan.

With respect to claim 1, Detlefs discloses a method of verifying with static checking whether a specified computer program satisfies a predefined set of conditions (see, for example, the title and page 24, Figure 10), comprising:

(a) converting the program into a logical equation representing the predefined set of conditions as applied to instructions and elements of the instructions of the program (see, for example, page 23, section 4, paragraph 1, lines 1-4, which shows generating a logical equation based on the conditions annotated in the program).

Although Detlefs discloses labeling any sub-equation of the logical equation (see, for example, page 29, section 6, paragraph 2, lines 1-2) with labels that identify source positions (lines 7-8), Detlefs does not expressly disclose the limitation wherein the converting step includes inserting flow control labels into

the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.

However, Chan discloses markers and scalars, i.e. labels, associated with conditional branches in a computer program for identifying paths of execution, i.e. the flow of control, through the program (see, for example, column 1, lines 38-53). In other words, Chan discloses flow control labels that identify conditional branch points in the specified computer program. The purpose is to check execution paths and verify the conditional branches (see, for example, column 1, lines 26-35).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the static checking and labeling of Detlefs with the labeling feature taught by Chan, for the purpose of verifying the conditional branches in the program. Therefore, in combination, Detlefs and Chan teach inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.

The combination of Detlefs and Chan further discloses:

(b) applying a theorem prover to the logical equation to determine the truth of the logical equation, and when the truth of the logical equation cannot be proved, generating at least one counter-example identifying one of the conditions, one or more variable values inconsistent with the one condition, and any of the flow control labels for conditional branch points of the program associated with

the identified variable values (see, for example, Detlefs, page 23, section 4, paragraph 1, lines 4-8, which shows using a theorem prover to determine when the truth of the logical equation does not hold, and page 29, section 6, paragraph 2, lines 3-7, which shows generating a counter-example and identifying both the sub-formulas and the associated labels that are false or inconsistent with the verification condition).

Note that the sub-formulas identified by Detlefs are inherently comprised of variable values (see, for example, page 30, which shows precondition formulas comprised of the variables "t" and "x" and the value "t.val," and which further shows the condition that variables have values). Therefore, in combination, Detlefs and Chan teach generating at least one counter-example identifying any of the flow control labels for conditional branch points of the program associated with the identified variable values.

The combination of Detlefs and Chan further discloses:

(c) converting the at least one counter-example into an error message that includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 3-4 and 7-9, which shows translating a counter-example into an error message that traces the source of the error based on the identified set of labels, and, Chan, column 1, lines 38-53, which shows that the markers and scalars, i.e. the labels, identify paths of execution through the computer program).

With respect to claim 2, the combination of Detlefs and Chan further discloses the limitation wherein the converting step additionally comprises a step of converting the program into an intermediate language form prior to creating the logical equation (see, for example, Detlefs, page 29, section 6, paragraph 1, lines 7-9, which shows converting the program into Dijkstra's guarded commands, i.e. an intermediate language, before generating the logical equation).

With respect to claim 3, the combination of Detlefs and Chan further discloses the limitation wherein the flow control labels are inserted before converting the program into the intermediate language form (see, for example, Detlefs, page 29, section 6, paragraph 1, lines 7-9, which shows converting an annotated program into an intermediate language; note that annotations are considered labels that are inserted before converting the program).

With respect to claim 4, the combination of Detlefs and Chan further discloses the limitation wherein the flow control labels are inserted after converting the program into the intermediate language form (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 6-7, which shows that the verification condition generator inserts labels into the logical equation; note that because the program is converted into the intermediate language form prior to generating the logical equation, the labels are thereby inserted after converting the program).

With respect to claim 5, the combination of Detlefs and Chan further discloses the limitation wherein the intermediate language form is Dijkstra's guarded command language (see, for example, page 29, section 6, paragraph 1, lines 7-9, which shows using Dijkstra's guarded commands).

With respect to claim 8, the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form L ==> P wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the flow control labels are for the mere purpose of identifying conditional branch points in the computer program, as recited in claim 1, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as L ==> P, or alternately with any characters so long as such labels perform the same function of identifying conditional branch

points in the computer program, as taught by Detlefs and Chan. See *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

With respect to claim 11, the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form L=K ==> P wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the flow control labels are for the purpose of identifying conditional branch points in the computer program, as recited in claim 1, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as L=K ==> P, or alternately with any characters so long as such labels perform the same function of identifying conditional branch points in the computer program, as taught by Detlefs and Chan. See *In re Bond*,

910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

With respect to claim 14, the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form {LBLPOS L P} wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the flow control labels are for the mere purpose of identifying conditional branch points in the computer program, as recited in claim 1, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as {LBLPOS L P}, or alternately with any characters so long as such labels perform the same function of identifying conditional branch points in the computer program, as taught by Detlefs and Chan. See *In re Bond*,

910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

With respect to claim 17, the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form ¬{LBLNEG L ¬P} wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the flow control labels are for the mere purpose of identifying conditional branch points in the computer program, as recited in claim 1, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as ¬{LBLNEG L ¬P}, or alternately with any characters so long as such labels perform the function of identifying conditional branch points in the computer program, as taught by Detlefs and Chan. See *In re*

Bond, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

With respect to claim 20, the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels is in the form {LBLPOS L True} ==> P wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the flow control labels are for the mere purpose of identifying conditional branch points in the computer program, as recited in claim 1, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as {LBLPOS L True} ==> P, or alternately with any characters so long as such labels perform the same function of identifying conditional branch points in the computer program, as taught by Detlefs and

Chan. See *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

With respect to claim 22, the combination of Detlefs and Chan further discloses the limitation wherein the flow control label name identifies a line number in the specified computer program at which an associated program instruction is located (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 7-8, which shows that the name of the label identifies a source position, i.e. a line number in the program) and includes a sequence number indicating an order of execution of the program instruction at the identified line number relative to other program instructions identified by other flow control labels (see, for example, Chan, column 1, lines 54-57, which shows sequence identifiers or numbers associated with the labels).

With respect to claim 23, the claim is analogous to claim 1 (therefore, see Detlefs and Chan as applied to claim 1 above). Note that Detlefs further discloses that the static checking system is implemented as a computer program product (see, for example, page 23, section 4, paragraph 2, lines 1-5). Also note that Detlefs discloses a verification condition generator, a theorem prover, control labels, and a post processor (see, for example, page 24, Figure 10, and page 29, section 6, paragraph 2, lines 1-4).

With respect to claim 24, see Detlefs and Chan as applied to claim 2 above.

With respect to claim 25, see Detlefs and Chan as applied to claim 3 above.

With respect to claim 26, see Detlefs and Chan as applied to claim 4 above.

With respect to claim 27, see Detlefs and Chan as applied to claim 5 above.

With respect to claim 30, see Detlefs and Chan as applied to claim 8 above.

With respect to claim 33, see Detlefs and Chan as applied to claim 11 above.

With respect to claim 36, see Detlefs and Chan as applied to claim 14 above.

With respect to claim 39, see Detlefs and Chan as applied to claim 17 above.

With respect to claim 42, see Detlefs and Chan as applied to claim 20 above.

With respect to claim 45, see Detlefs and Chan as applied to claim 22 above.

With respect to claim 47, Detlefs discloses a method, comprising:

(a) converting a computer program into a logical equation representing a predefined set of conditions (see, for example, page 23, section 4, paragraph 1, lines 1-4, which shows generating a logical equation based on the conditions annotated in a computer program);

Although Detlefs discloses labeling any sub-equation of the logical equation (see, for example, page 29, section 6, paragraph 2, lines 1-2) with labels that identify source positions (lines 7-8), Detlefs does not expressly disclose:

(b) inserting flow control labels into sub-equations of the logical equation, the flow control labels identifying branch points in the computer program, wherein at least one of the flow control labels is of a form selected from the group consisting of (1) L ==> P wherein L is a flow control label name and P is a subcomponent of the logical equation, (2) L=K ==> P wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation, (3) {LBLPOS L P} wherein L is a flow control label name and P is a subcomponent of the logical equation, (4) ¬{LBLNEG L ¬P} wherein L is a flow control label name and P is a subcomponent of the logical equation, and (5) {LBLPOS L True} ==> P wherein L is a flow control label name and P is a subcomponent of the logical equation.

However, Chan discloses markers and scalars, i.e. labels, associated with conditional branches in a computer program for identifying paths of execution, i.e. the flow of control, through the program (see, for example, column 1, lines

38-53). In other words, Chan discloses flow control labels that identify conditional branch points in the specified computer program. The purpose is to check execution paths and verify the conditional branches (see, for example, column 1, lines 26-35).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to supplement the static checking and labeling of Detlefs with the labeling feature taught by Chan, for the purpose of verifying the conditional branches in the program. Therefore, in combination, Detlefs and Chan teach inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program.

Furthermore, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the flow control labels are for the mere purpose of identifying branch points in the computer program, as recited in the claim, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as L ==> P, L=K ==> P, {LBLPOS L P}, ¬{LBLNEG L ¬P}, or {LBLPOS L True} ==> P, or alternately with any characters so long as such labels perform the same function of identifying branch points in the computer program, as taught by Detlefs and Chan. See *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

The combination of Detlefs and Chan further discloses:

(c) applying a theorem prover to the logical equation to determine a truth of the logical equation, and when the truth of the logical equation cannot be proved, generating at least one counter-example identifying one of the conditions (see, for example, Detlefs, page 23, section 4, paragraph 1, lines 4-8, which shows using a theorem prover to determine when the truth of the logical equation does not hold, and page 29, section 6, paragraph 2, lines 3-7, which shows generating a counter-example and identifying both the sub-formulas and the associated labels that are false or inconsistent with the verification condition); and

(d) converting the at least on counter-example into an error message that comprises a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 3-4 and 7-9, which shows translating a counter-example into an error message that traces the source

of the error based on the identified set of labels, and, Chan, column 1, lines 38-53, which shows that the markers and scalars, i.e. the labels, identify paths of execution through the computer program).

3. Claims 6, 7, 9, 10, 12, 13, 15, 16, 18, 19, 21, 28, 29, 31, 32, 34, 35, 37, 38, 40, 41, 43, 44 and 46 are rejected under 35 U.S.C. 103(a) as being unpatentable over Detlefs in view of Chan, as applied to claims 1, 8, 11, 14, 17, 20, 23, 30, 36, 39 and 42 above, respectively, and further in view of Rickel.

With respect to claim 6, the combination of Detlefs and Chan further discloses the limitation wherein at least one of the flow control labels includes a flow control label name that includes a string that identifies a line number in the specified computer program (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 7-8, which shows that the name of the label includes a source position, i.e. a line number in the program; note that a label name is inherently a string).

Although the combination of Detlefs and Chan discloses that the string identifies a type of error (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 7-8), the combination of Detlefs and Chan does not expressly disclose the limitation wherein the string identifies a type of branch in the program.

However, Rickel discloses branch labels in an intermediate language form of a program (see, for example, column 8, lines 20-23) used to represent every flow path in the program based on a branch type such as a call or a jump (see, for

example, column 8, lines 47-53), for the purpose of traversing all the paths in a program with a static debugger and finding any errors (see, for example, column 8, line 61 to column 9, line 3).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the static checking and labeling of Detlefs and Chan with the labeling feature taught by Rickel, for the purpose of checking all the paths in the computer program and finding any errors.

With respect to claim 7, although the combination of Detlefs and Chan discloses a flow control label name that identifies a line number in the specified computer program (see, for example, Detlefs, page 29, section 6, paragraph 2, lines 7-8, which shows that the name of the label identifies a source position, i.e. a line number in the program), the combination of Detlefs and Chan does not expressly disclose the limitation wherein at least one of the flow control labels includes a flow control label name that includes a value associated with an entry in a table that identifies a type of branch in the program and a line number in the specified computer program.

However, Rickel discloses branch labels in an intermediate language form of a program (see, for example, column 8, lines 20-23) used to represent every flow path in the program based on a branch type such as a call or a jump (see, for example, column 8, lines 47-53). Rickel further discloses using jump tables and jump targets from the symbol table as the branch labels (see, for example, FIG. 7).

The purpose is to traverse all the paths in a program with a static debugger and find any errors (see, for example, column 8, line 61 to column 9, line 3).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the static checking and labeling of Detlefs and Chan with the labeling feature taught by Rickel, for the purpose of checking all the paths in the computer program and finding any errors.

With respect to claims 9, 12, 15, 18 and 21, see Detlefs, Chan and Rickel as applied to claim 6 above.

With respect to claims 10, 13, 16, 19 and 46, see Detlefs, Chan and Rickel as applied to claim 7 above.

With respect to claim 28, 31, 37, 40, 43 and 34, see Detlefs, Chan and Rickel as applied to claim 6 above.

With respect to claim 29, 32, 38, 41, 44 and 35, see Detlefs, Chan and Rickel as applied to claim 7 above.

(11) Response to Argument

A. Group I (claim 1)

"Detlefs and/or Chan not teach/suggest Tracing Path Limitation" (pages 5-6)

Appellant contends that the limitation, "includes a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow

control labels," is not taught or suggested in Detlefs and/or Chan (Appellant's argument, page 5, first paragraph). Specifically, Appellant contends that the combination of Detlefs and Chan fails to teach or suggest an error message that includes a program trace that identifies a path through the computer program (Appellant's argument, page 6, second paragraph).

However, Detlefs teaches error messages that identify source positions in the computer program based on the labels identified in the counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-4 and 7-9). Thus, Detlefs teaches an error message including a program trace that identifies positions in the computer program when the counter-example identifies one or more of the labels.

Chan teaches detecting and verifying conditional branches along the execution path of the computer program (see, for example, column 1, lines 26-35). Chan further teaches markers and scalars for identifying paths of execution in the computer program (see, for example, column 1, lines 38-53).

Therefore, in combination, Detlefs and Chan teach an error message including a program trace that identifies a path through the computer program when the counter-example identifies one or more of the flow control labels.

Moreover, although Appellant characterizes Detlefs as reporting "an exact single location of an error message" (Appellant's argument, page 6, second paragraph), in contrast, Detlefs expressly discloses reporting error messages based on "a set of labels" wherein the name of a label, i.e. each label, "encodes the source position and error type" (see page 29, section 6, paragraph 2, lines 3-8).

"Detlefs and/or Chan not teach/suggest Flow Control Label Limitation" (pages 6-7)

Appellant contends that the limitation, "inserting flow control labels into the sub-equations of the logical equation, the flow control labels identifying conditional branch points in the specified computer program," is not taught or suggested in Detlefs and/or Chan (Appellant's argument, page 6, third paragraph).

However, Detlefs teaches a method of static checking (see, for example, the title) that comprises a logical equation derived from a computer program (see, for example, page 23, section 4, paragraph 1, lines 1-4). Detlefs further teaches labeling any sub-formula or sub-equation of the logical equation (see, for example, page 29, section 6, paragraph 2, lines 1-2) with labels that identify source positions (lines 7-8). Thus, Detlefs teaches inserting labels into the sub-equations of the logical equation, such that the labels identify locations in the specified computer program.

Chan teaches markers and scalars associated with conditional branches in a computer program for identifying paths of execution, i.e. the flow of control, through the program (see, for example, column 1, lines 38-53). Thus, Chan teaches flow control labels (markers and scalars) that identify conditional branch points in the specified computer program.

Therefore, in combination, Detlefs and Chan teach inserting flow control labels into the sub-equations of the logical equation, such that the flow control labels identify conditional branch points in the specified computer program.

Appellant characterizes the above combination of Detlefs and Chan as teaching a dynamic checker that, at runtime, reports the location of specific error messages (Appellant's argument, page 7, first paragraph). However, Detlefs teaches a static checker, as acknowledged by Appellant. The static checker of Detlefs is supplemented with the flow control labels

suggested by Chan. Moreover, Chan discloses, "a global label (path identifier) is stored at compile time which computes a check marker ..." (see the abstract).

"Detlefs and/or Chan not teach/suggest Theorem Prover Limitation" (pages 7-8)

Appellant contends that the limitation, "generating ... any of the flow control labels for conditional branch points of the program associated with the identified variable values," is not taught or suggested in Detlefs and/or Chan (Appellant's argument, page 7, second paragraph).

However, Detlefs teaches a theorem prover for evaluating the verification condition (see, for example, page 23, section 4, paragraph 1, lines 4-8). Detlefs further teaches generating a counter-example and identifying both the sub-formulas and the associated labels that are false or inconsistent with the condition (see, for example, page 29, section 6, paragraph 2, lines 3-7). Such sub-formulas identified by Detlefs are inherently comprised of variable values (see, for example, page 30, which shows precondition formulas comprised of the variables "t" and "x" and the value "t.val"). Detlefs further discloses, "the language-enforced condition that variables have values of their declared types is important not only for assumed preconditions ..." (see page 30). Thus, Detlefs teaches generating a counter-example with labels for points of the program associated with the identified variable values.

Chan teaches markers and scalars associated with conditional branches in a computer program for identifying paths of execution, i.e. the flow of control, through the program (see, for example, column 1, lines 38-53). Thus, Chan teaches flow control labels (markers and scalars) that identify conditional branch points in the specified computer program.

Therefore, in combination, Detlefs and Chan teach generating a counter-example with flow control labels for conditional branch points of the program associated with the identified variable values.

B. Group II (claim 47)

"Detlefs and/or Chan not teach/suggest Tracing Path Limitation" (pages 9-10)

Again, Appellant's argument is directed toward the same issue(s) as in Group I, which has been addressed above (see pages 20-21 of this Answer).

"Detlefs and/or Chan not teach/suggest Flow Control Labels Limitation" (pages 10-11)

Appellant contends that the limitation, "selected from the group consisting of (1) L ==> P wherein L is a flow control label name and P is a subcomponent of the logical equation, (2) L=K ==> P wherein L is a flow control label name, k is a constant value and P is a subcomponent of the logical equation, (3) {LBLPOS L P} wherein L is a flow control label name and P is a subcomponent of the logical equation, (4) ¬{LBLNEG L ¬P} wherein L is a flow control label name and P is a subcomponent of the logical equation, and (5) {LBLPOS L True} ==> P wherein L is a flow control label name and P is a subcomponent of the logical equation," is not taught or suggested by Detlefs and/or Chan (Appellant's arguments, page 10, third paragraph).

However, Detlefs discloses using Dijkstra's guarded commands and weakest-precondition equations to generate a verification condition (see, for example, page 29, section 6, paragraph 1, lines 7-9). Detlefs further discloses using labels in the verification condition, and

using a theorem prover to determine the truth of the logical equation and to find a counter-example (see, for example, page 29, section 6, paragraph 2, lines 1-9).

It should be noted that the claim recites, "the flow control labels identifying branch points in the computer program, wherein at least one of the flow control labels is of a form selected from the group consisting of (1) L ==> P ..." (claim 47). Thus, the flow control labels are for the mere purpose of identifying branch points in the computer program, regardless of the particular form or notation. Accordingly, the particular form or notation of such labels has not been given much patentable weight.

As presented above, in combination, Detlefs and Chan provide a means to insert flow control labels into the sub-equations of the logical equation, such that the flow control labels identify conditional branch points in the specified computer program.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to implement the labeling of Detlefs and Chan with a form or notation such as L ==> P, L=K ==> P, {LBLPOS L P}, ¬{LBLNEG L ¬P}, or {LBLPOS L True} ==> P, or alternately with any characters so long as such labels perform the same function of identifying branch points in the computer program, as taught by Detlefs and Chan. See *In re Bond*, 910 F.2d 831, 15 USPQ2d 1566 (Fed. Cir. 1990). Thus, the combination of Detlefs and Chan reads on the claimed limitations.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

MY

Michael J. Yigdall
Examiner
Art Unit 2122

mjy
November 23, 2004

Conferees
Tuan Dam, SPE 2122
Kakali Chaki, SPE 2124

TUAN DAM
SUPERVISORY PATENT EXAMINER

Kakali Chaki.
KAKALI CHAKI
SUPERVISORY PATENT EXAMINER
TECHNOLOGY CENTER 2100

PENNIE & EDMONDS LLP
3300 Hillview Avenue
Palo Alto, CA 94304